浙江大学
ZHEJIANG UNIVERSITY

Computer Architecture Experiment

Topic 3. Pipelined CPU with forwarding

浙江大学计算机学院
陈文智
chenwz@zju.edu.cn

实验操作流程

- 阅读实验文档，理解支持Forwarding的流水线处理器的实现方式
- 以前一次实验为基础，为处理器增加Forwarding功能
- 对处理器进行仿真，检验处理器的仿真结果是否符合要求。
- 综合工程并下载至开发板，在单步执行的过程中检查调试屏幕的输出，检验处理器的执行过程是否正确。

实验验收标准

- 仿真执行过程中，处理器的行为和内部控制信号均符合要求。
- 下载至开发板后的单步执行过程中，寄存器的变化过程和最终执行结果与测试程序相吻合。

Outline

- Experiment Purpose
- Experiment Task
- Basic Principle
- Operating Procedures
- Precaution
- Checkpoints

Experiment Purpose

- Understand the principles of Pipelined CPU Bypass Unit
- Master the method of Pipelined Pipeline Forwarding Detection and Pipeline Forwards.
- Master the Condition In Which Pipeline Forwards.
- master methods of program verification of Pipelined CPU with forwarding

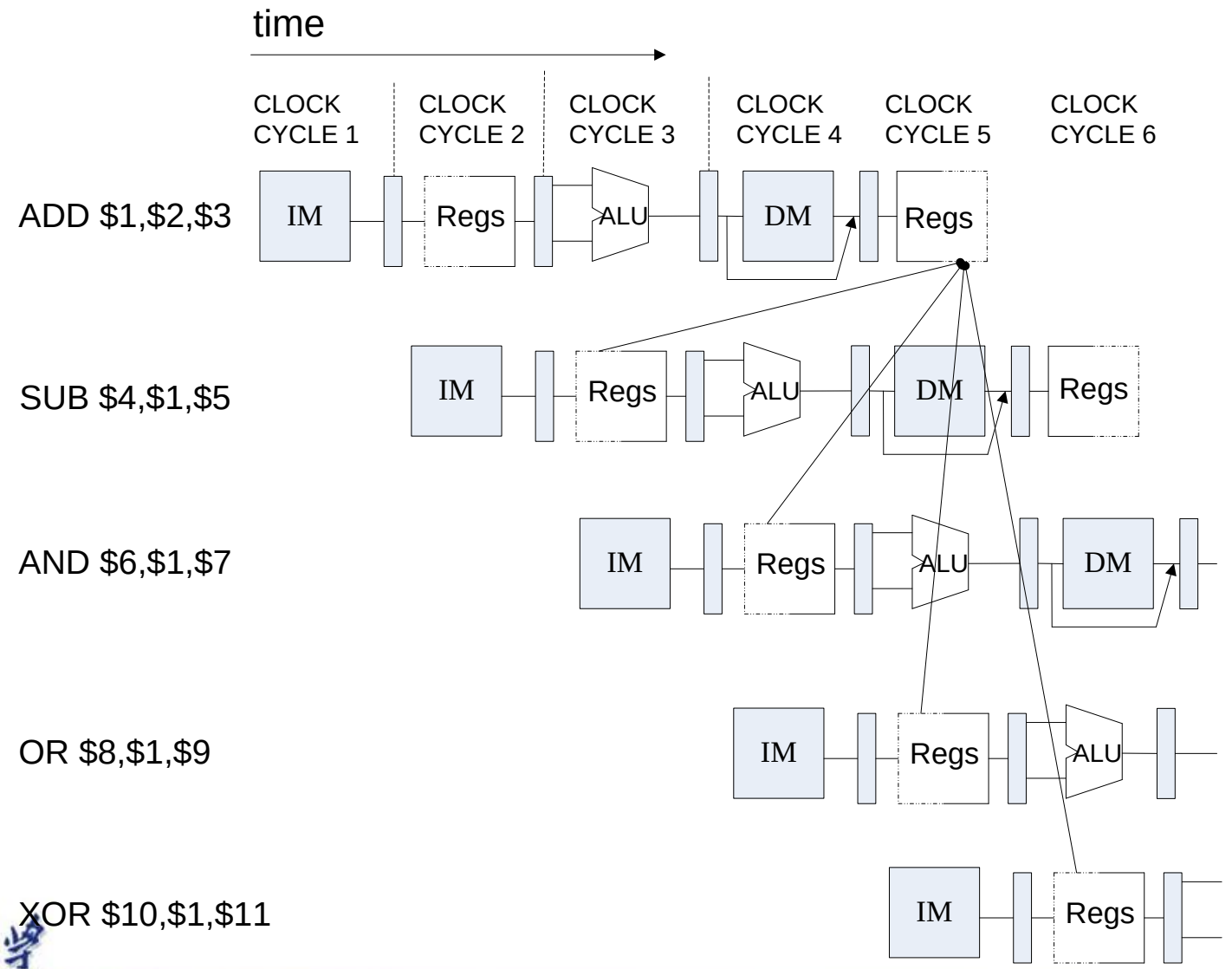
Experiment Task

- **Design the Bypass Unit of Datapath of 5-stages Pipelined CPU**
- **Modify the CPU Controller**
  - Conditions in Which Pipeline Forwards.
- **Verify the Pipeline CPU with program and observe the execution of program**

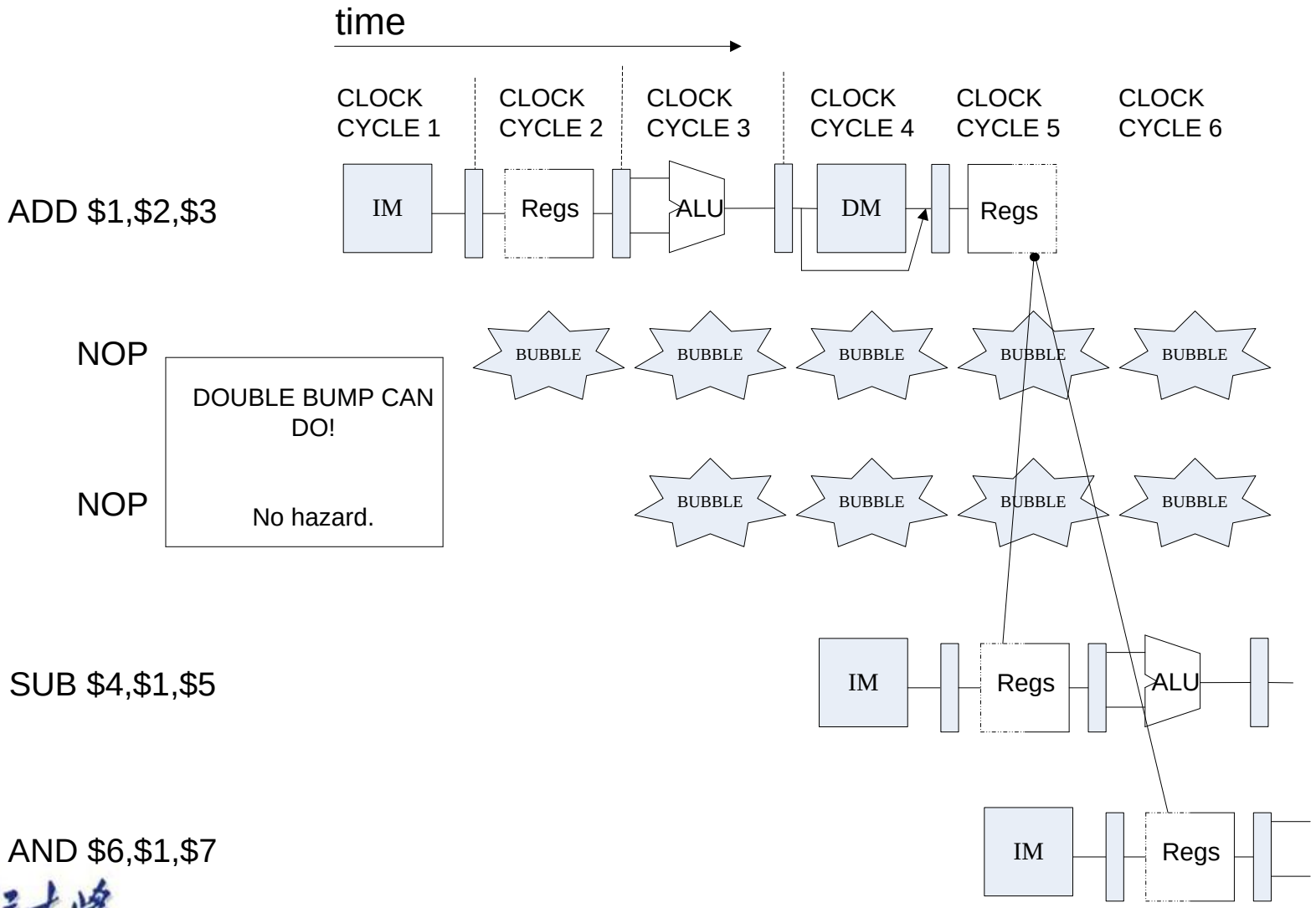
Data Hazard Stalls

- **Minimizing Data Hazard Stalls by Forwarding: In most cases, the problem can be resolved by forwarding, also called bypassing, short-circuiting.**
- **Data Hazards Requiring Stalls: In some cases, data hazards can NOT be handled by bypassing.(Next topic)**

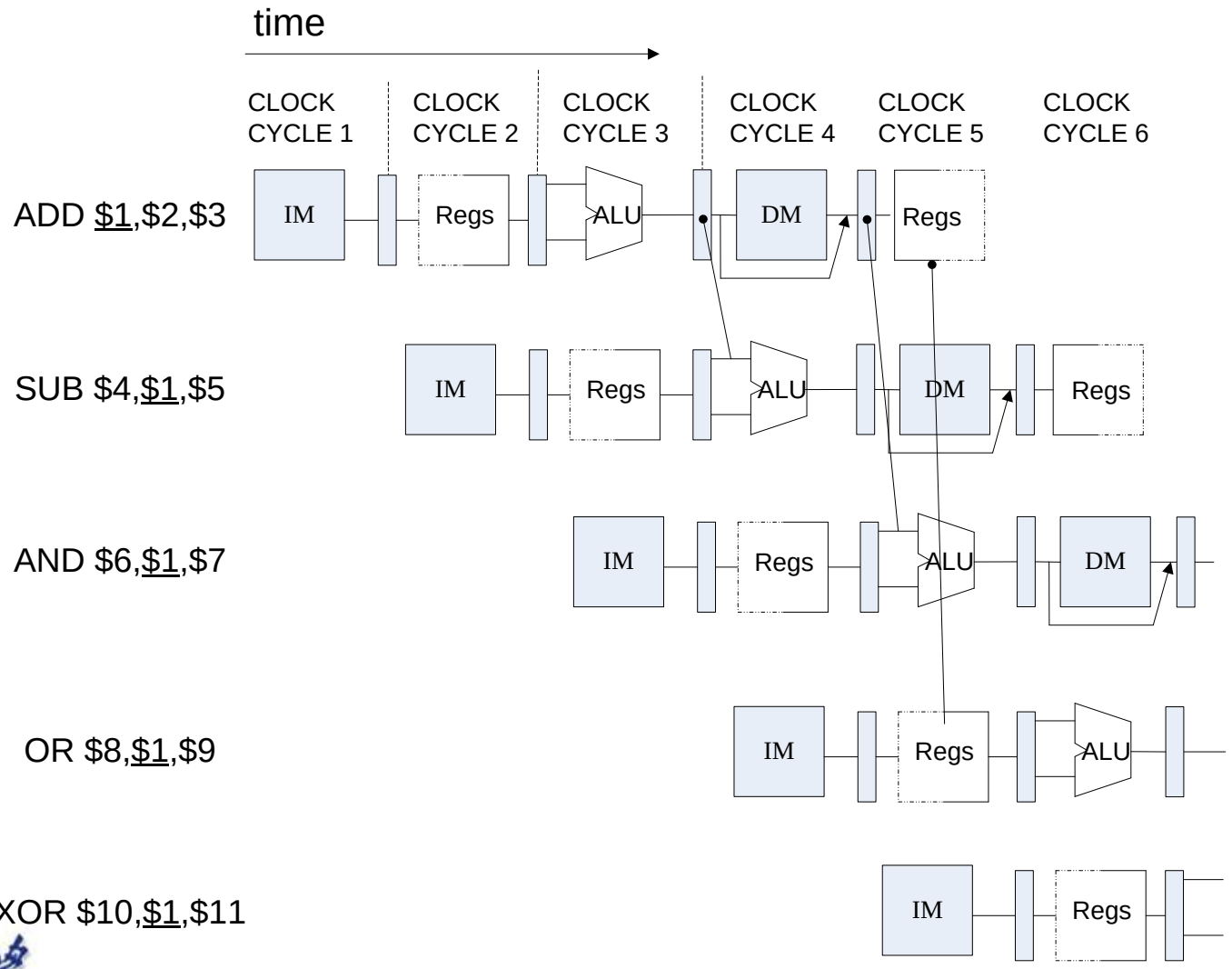
Instruction Demo



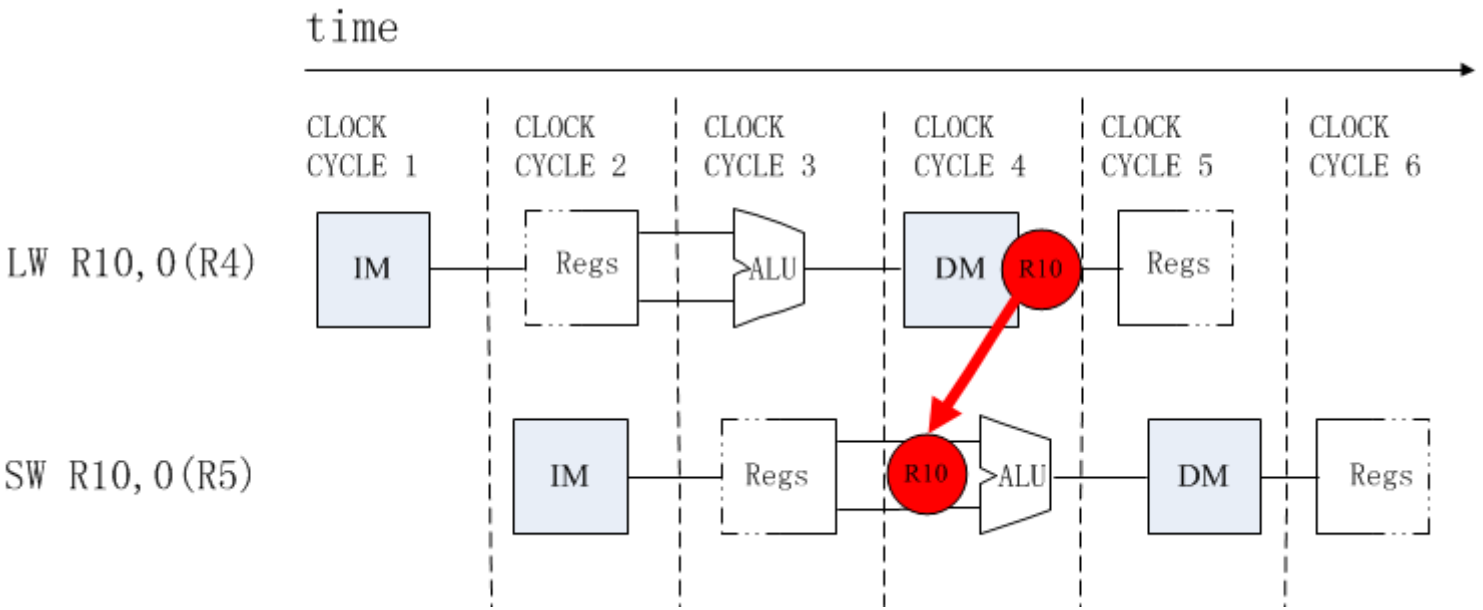
Data Hazard Causes Stalls



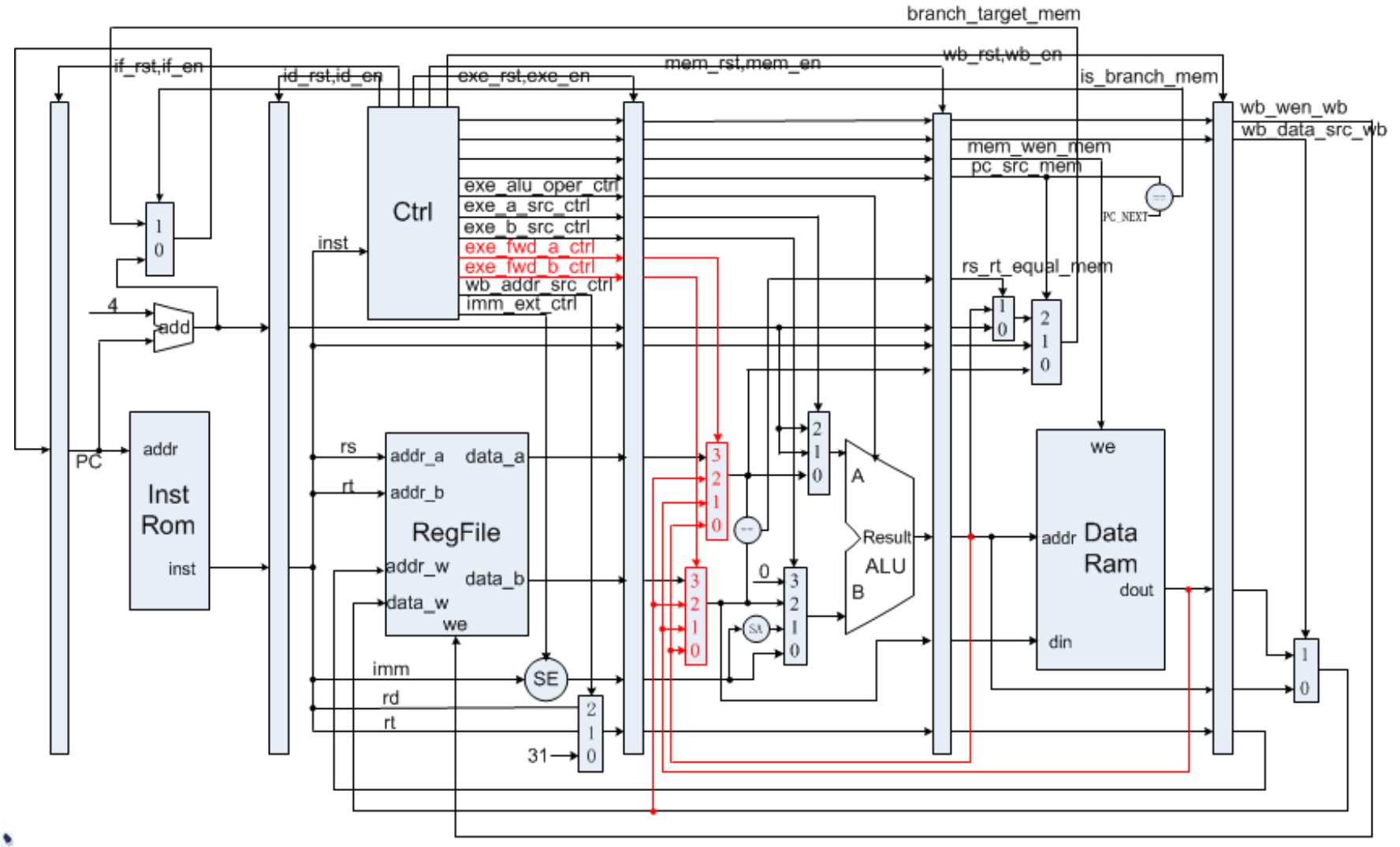
Pipeline Forward to Avoid the Data hazard



R/I/J-Type After LW



Datapath with Bypass Unit



Controller

```
if(MEM.WriteReg and (MEM.RegisterRd!=0)
    and(MEM.RegisterRd==EXE.RegisterRs)) ForwardA=01
if(MEM.WriteReg and (MEM.RegisterRd!=0)
    and(MEM.RegisterRd==EXE.RegisterRt)) ForwardB=01
if(MEM.WriteReg and (MEM.RegisterRd!=0)
    and(MEM.RegisterRd==EXE.RegisterRs) and MEM.ReadMEM) ForwardA=10
if(MEM.WriteReg and (MEM.RegisterRd!=0)
    and(MEM.RegisterRd==EXE.RegisterRt) and MEM.ReadMEM) ForwardB=10
if(WB.WriteReg and (WB.RegisterRd!=0)
    and(MEM.RegisterRd!=EXE.RegisterRs)
    and(WB.RegisterRd==EXE.RegisterRs)) ForwardA=11
if(WB.WriteReg and (WB.RegisterRd!=0)
    and(MEM.RegisterRd!=EXE.RegisterRt)
    and(WB.RegisterRd==EXE.RegisterRt)) ForwardB=11
```

Program for verification (1)

```
00000824 main:    and $1, $0, $0      # address of data[0]
34240050          ori $4, $1, 80      # address of data[0]
20050004 call:    addi $5, $0, 4      # counter
0c00000a          jal sum             # call function
ac820000 return:  sw $2, 0($4)        # store result
8c890000          lw $9, 0($4)        # check sw
ac890004          sw $9, 4($4)        # store result again
01244022          sub $8, $9, $4      # sub: $8 <- $9 - $4
08000008 finish:  j finish            # dead loop
00000000          nop                 # done
```

Program for verification (2)

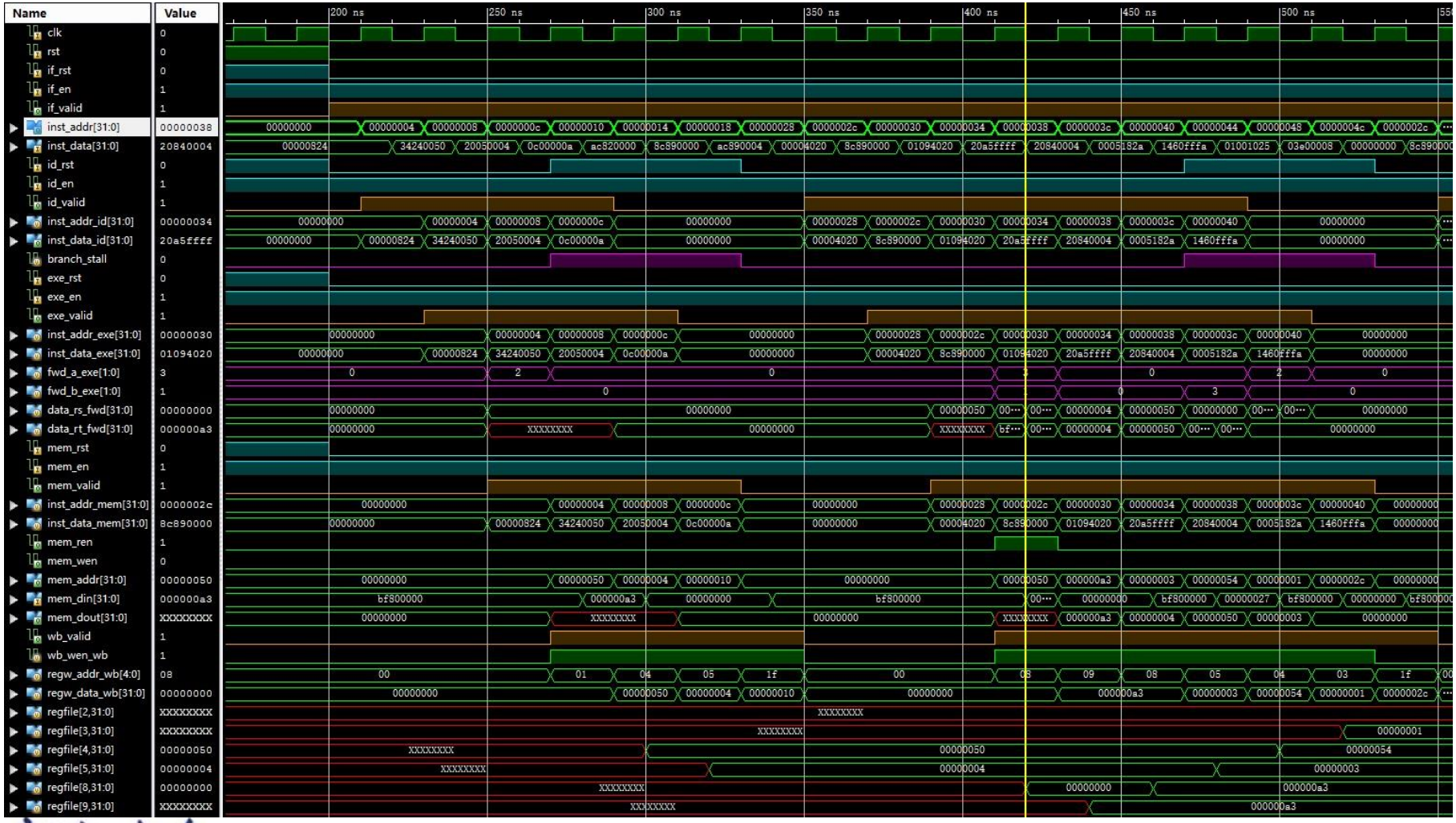
```
00004020 sum:   add $8, $0, $0      # sum function entry
8c890000 loop:  lw $9, 0($4)        # load data
01094020        add $8, $8, $9      # sum
20a5ffff        addi $5, $5, -1     # counter - 1
20840004        addi $4, $4, 4      # address + 4
0005182a        slt $3, $0, $5      # finish?
1460fffa        bne $3, $0, loop    # finish?
01001025        or $2, $8, $0       # move result to $v0
03e00008        jr $ra              # return
00000000        nop                 # done
```

Checkpoints

- **CP 1: Waveform Simulation of Pipelined CPU with forwarding**

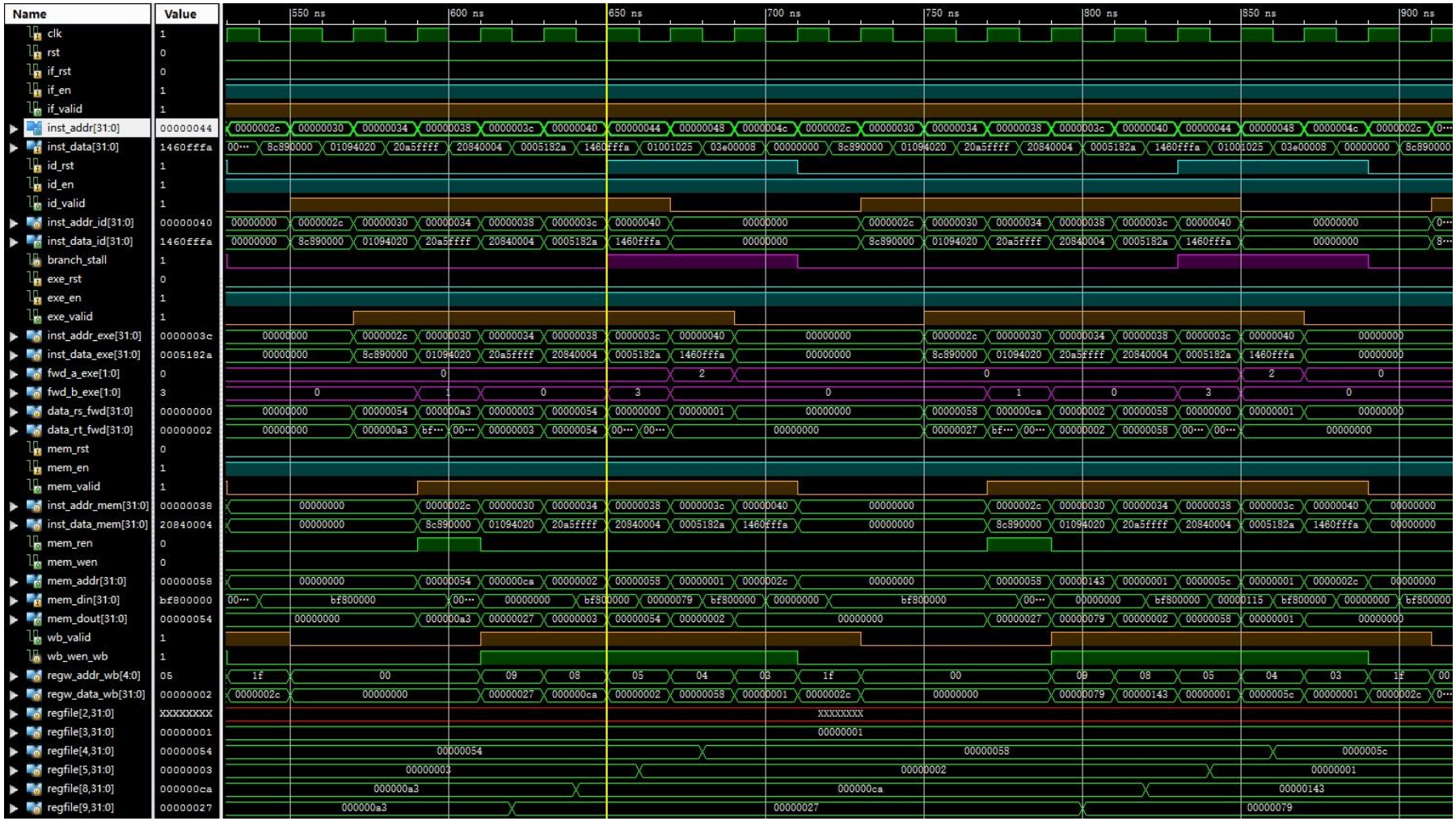
- **CP 2: FPGA Implementation of Pipelined CPU with forwarding with the verification program**

Simulation (1)

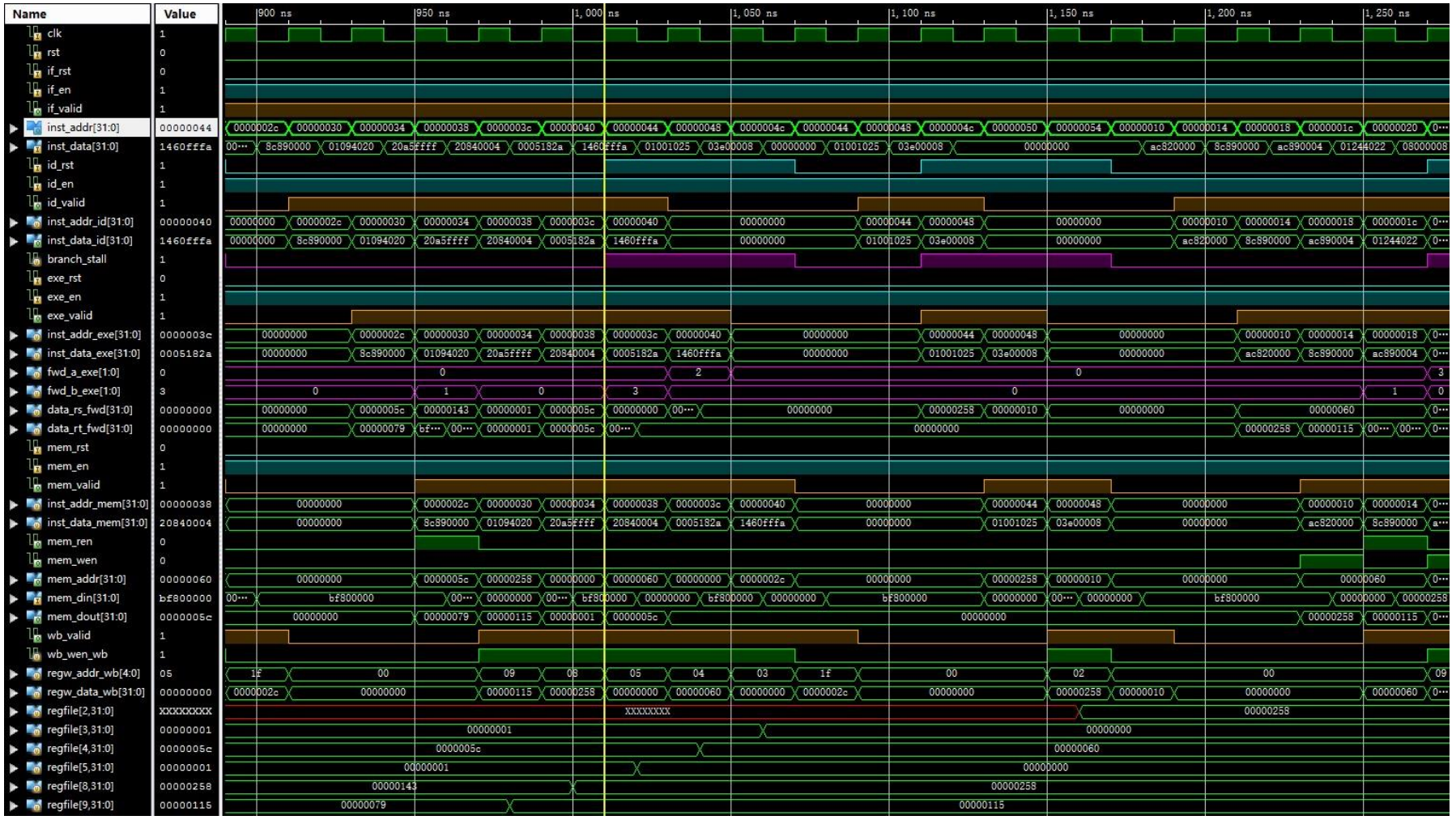


浙江大学 ZheJiang University

Simulation (2)



Simulation (3)



Thanks!

浙江大学
ZheJiang University